LAU
الجامعة اللبنانية الأميركية
Lebanese American University

CSC 498R: Internet of Things

Lecture 10: A Quick Intro to Machine Learning

Instructor: Haidar M. Harmanani

Fall 2017

Field of study that gives computers the ability to learn without being explicitly programmed

Arthur Samuel

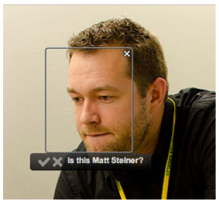
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Tom Mitchell

What is Machine Learning?

What is Machine Learning?

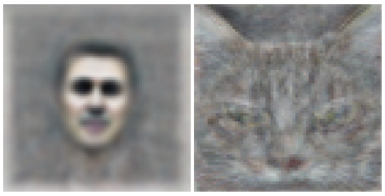
- More than just memorizing data
- Fundamentally, learning is generalization
  - Find the underlying structure of training data
- Two broad classes of example use
  - Prediction (continuous): "How much snow are we going to have tomorrow?"
  - Classification (discrete): face recognition



Machine Learning Today

Microsoft translates English to Chinese in real-time

Google learns face/cat detector by feeding machines tons of unlabeled images



Facebook recognizes human faces and the identities

Machine Learning Tools

LIBSVM -- A Library for Support Vector Machines

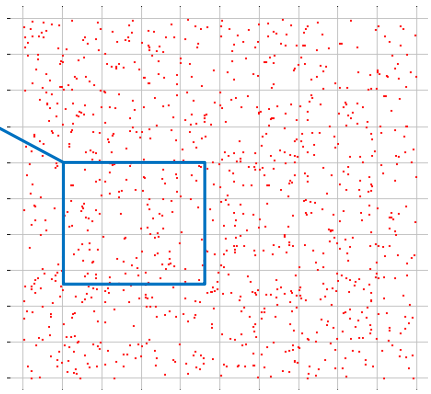
LIBLINEAR -- A Library for Large Linear Classification

TensorFlow

SPArse Modeling Software
now open-source!

Predicting from Samples

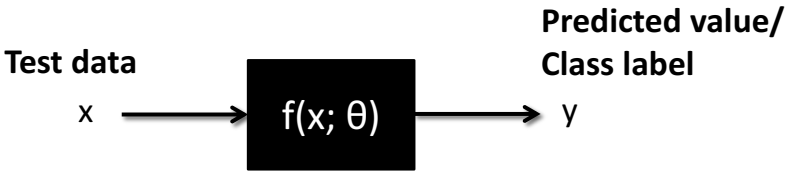
- Most datasets are samples from an infinite population.
- We are most interested in models of the population, but we have access only to a sample of it.



- For datasets consisting of (X,y)
  - features X + label y
  - A model is a prediction y = f(X)
- Train on a training sample D and we denote the model as $f_D(X)$

Idea: Build a Model that Maps Your Data to "Labels"

- Use a (parametric) model
  - Define a black box using a set of parameters **θ**
  - This black box should read in a data vector and output the prediction (continuous value or class label)
  - Formally, y = f(x; **θ**)



- "Learning" is to determine the parameters **θ** using a set of training data

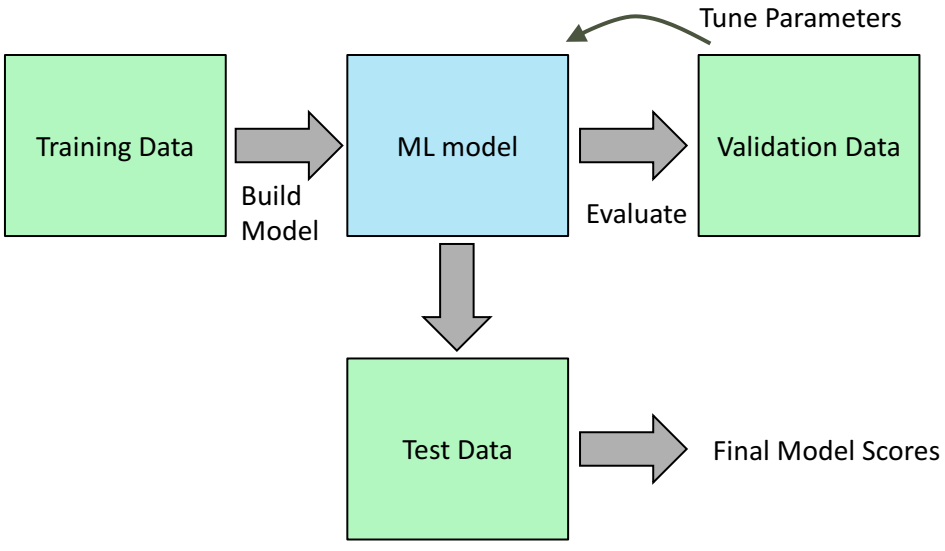
Train-Test-Validation Sets

- When making measurements on a ML algorithm, we have additional challenges.
- With a sample of data, any model fit to it models both:
  - Structure in the entire population
  - Structure in the specific sample not true of the population

Train-Test-Validation Sets

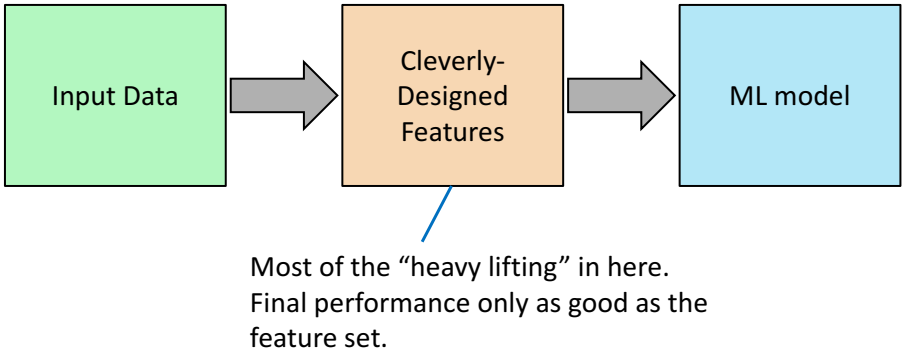
- Example: a 25-year old man and a 30-year old woman.
  - Age predicts gender perfectly. (age < 27 => man else woman)
  - Gender predicts age perfectly. (gender == man => 25 else 30)
  - Neither result generalizes. This is called over-fitting.

Model Tuning



A Brief History of Machine Learning

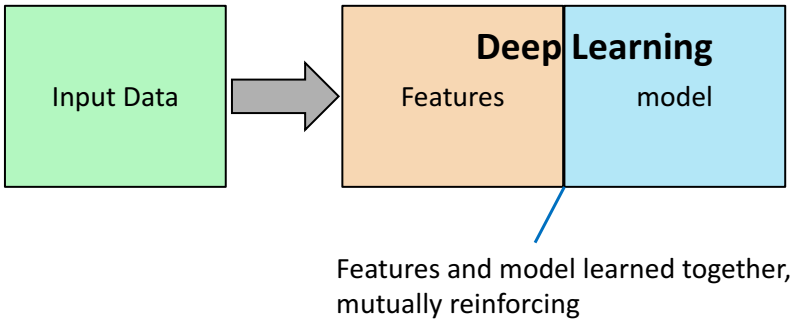
- Before 2012*:



- * Before publication of Krizhevsky et al.'s ImageNet CNN paper.

A Brief History of Machine Learning

- After 2012:



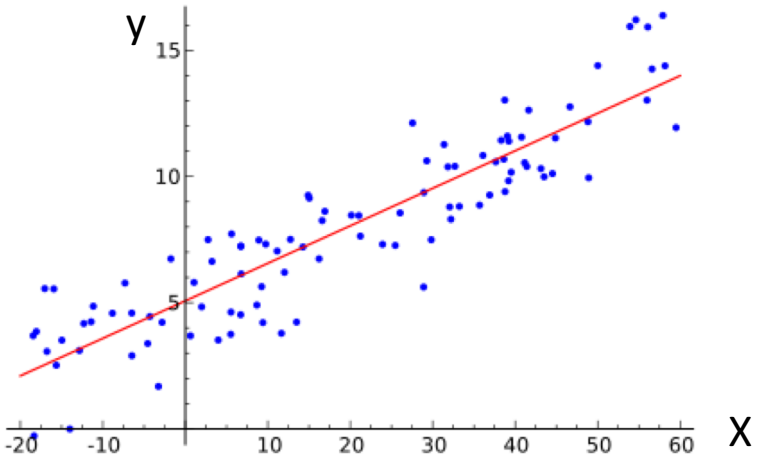
A Brief History of Machine Learning

- But this (pre-2012) picture is still typical of many pipelines.
- We'll focus on one aspect of feature design: feature selection, i.e. choosing which features from a list of candidates to use for a ML problem.

Choosing the Right Model

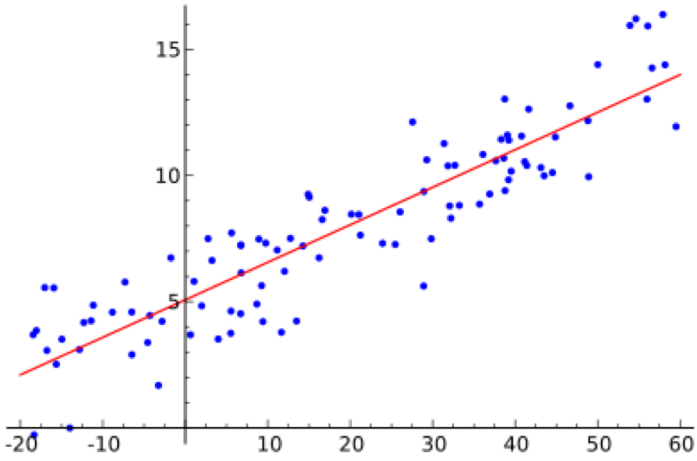
- Should have some insights on why this model would fit the data well or make correct class predictions
- Keep the model simple (i.e., not too many parameters)
  - So you have enough data to train
  - So you can train in a reasonable amount of time

Predicting from Samples: Example 1

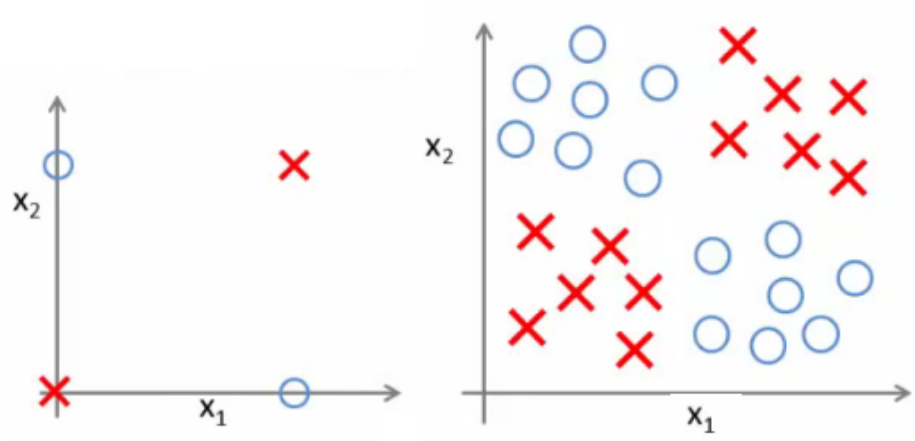


Predicting from Samples: Example 1

- Easy: Find the best line (linear function y=f(X)) to explain the data.

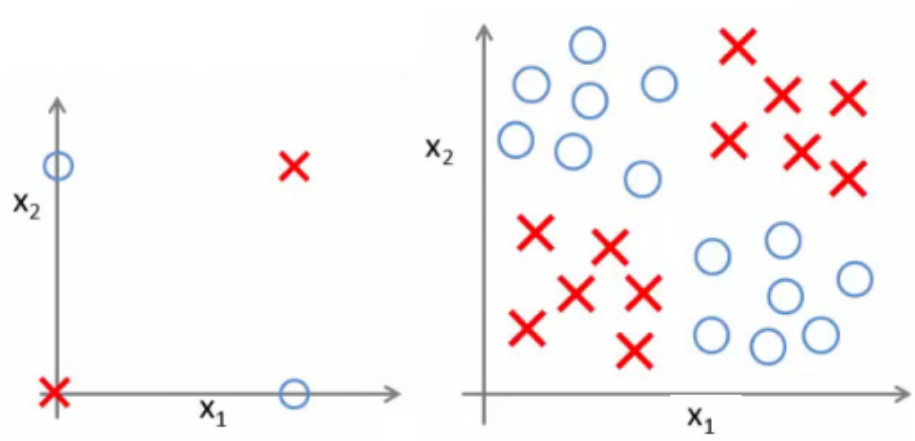


Predicting from Samples: Example 2



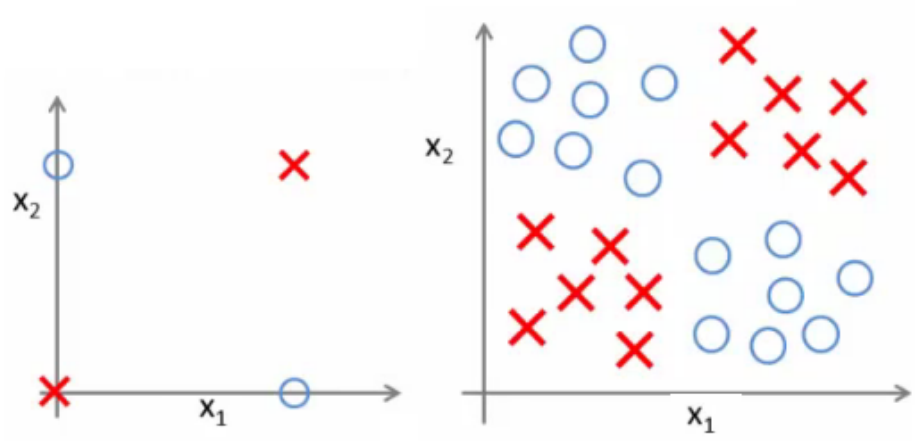
Predicting from Samples: Example 2

- Tough using Logistic Regression
  - Although doable



Predicting from Samples: Example 2

- Solution?
  - Use non-linear approaches but sophisticated ones will wait till next time!

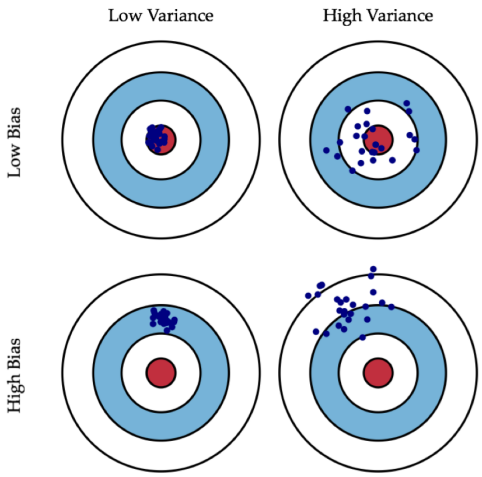


Bias and Variance

- Our data-generated model $f_D(X)$ is a statistical estimate of the true function $f(X)$.
- Because of this, its subject to bias and variance

Bias and Variance

- Bias: if we train models $f_D(X)$ on many training sets D, bias is the expected difference between their predictions and the true y's.
  - $Bias = \mathrm{E}[f_D(X) - y]$
  - E[] is taken over points X and datasets D
- Variance: if we train models $f_D(X)$ on many training sets D, variance is the variance of the estimates:
  - $Variance = \mathrm{E}\left[\left(f_D(X) - \bar{f}(X)\right)^2\right]$
  - Where $\bar{f}(X) = \mathrm{E}[f_D(X)]$ is the average prediction on X.

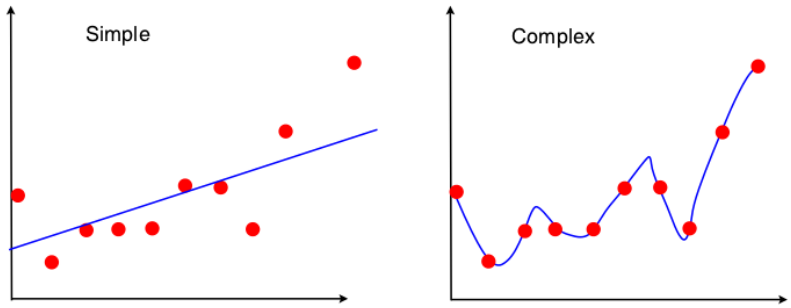


Bias and Variance Tradeoff

- There is usually a bias-variance tradeoff caused by model complexity.
- Complex models (many parameters) usually have lower bias, but higher variance.
- Simple models (few parameters) have higher bias, but lower variance.

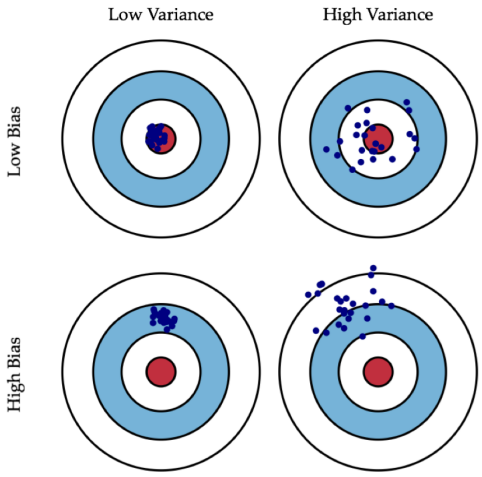
Bias and Variance Tradeoff: Example

- A linear model can only fit a straight line while a high-degree polynomial can fit a complex curve.
- The polynomial can fit the individual sample, rather than the population.
  - Its shape can vary from sample to sample, so it has high variance.



Bias and Variance Tradeoff

- The total expected error is $Bias^2 + Variance$
- Because of the bias-variance trade-off, we want to balance these two contributions.
- If $Variance$ strongly dominates, it means there is too much variation between models.
  - Over-fitting.
- If $Bias$ strongly dominates, then the models are not fitting the data well enough
  - Under-fitting.



A model that has been over-fitted has poor predictive performance, as it overreacts to minor fluctuations in the training data

Under-fitting would occur, for example, when fitting a linear model to non-linear data. Such a model would have poor predictive performance.

Bias and Variance Tradeoff

Back to Machine Learning …

- Lots of data that are generated by a lot of sources
  - Want techniques that minimize software engineering effort
  - simple algorithms, teach computer how to learn from data
  - don't spend time hand-engineering algorithms or high level features from the raw data

Machine Learning

- Learning by labeled example: *supervised learning*
  - e.g. An email spam detector
  - amazingly effective if you have lots of examples
- Discovering patterns: *unsupervised learning*
  - e.g. data clustering
  - difficult in practice, but useful if you lack labeled examples
- Feedback right/wrong: *reinforcement learning*
  - e.g. learning to play chess by winning or losing
  - works well in some domains, becoming more important

Machine Learning: Supervised

- Given data about the size of houses on the real estate market, try to predict their price.
  - Price as a function of size is a continuous output, so this is a regression problem.
- Other Examples:
  - Is this image a cat, dog, car, house?
  - How would this user score that restaurant?
  - Is this email spam?
  - Is this blob a supernova?

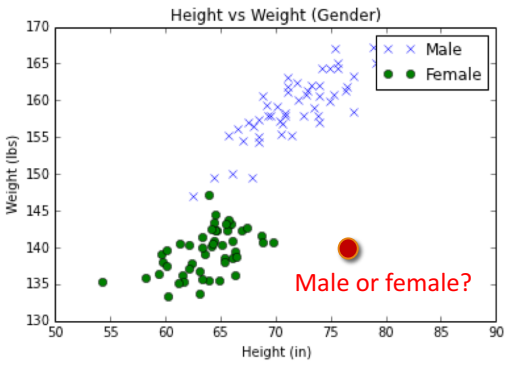
Machine Learning: Unsupervised

- Clustering
  - Cluster some hand-written digit data into 10 classes.
  - Take a collection of 1000 essays written on the US Economy, and find a way to automatically group these essays into a small number that are somehow similar or related by different variables, such as word frequency, sentence length, page count, and so on.
  - What are the top 20 topics in `Twitter` right now?
  - Find and cluster distinct accents of people in a location
  - Cluster News based on topics

- Associative
  - A doctor over years of experience forms associations in his mind between patient characteristics and illnesses that they have
  - If a new patient shows up then based on this patient's characteristics such as symptoms, family medical history, physical attributes, mental outlook, etc the doctor associates possible illness or illnesses based on what the doctor has seen before with similar patients.

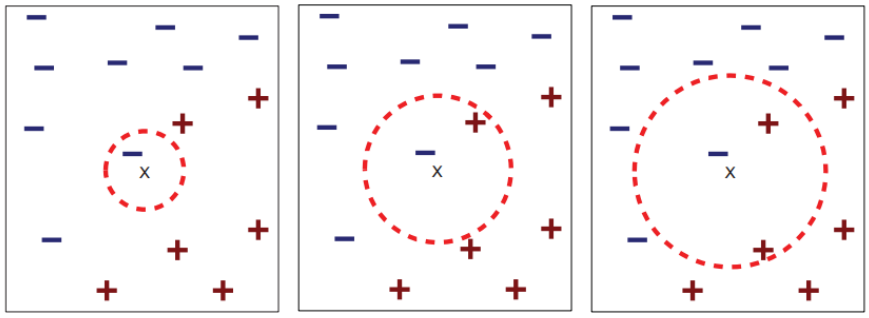
Classification

- Given a new sample, predicting discrete values (the class) instead of continuous values
- Several popular models
  - Nearest neighbor
  - Decision tree
  - Support vector machine



k-Nearest Neighbors

- Memorize all training samples, search for the closest ones to the test sample
- Classification by majority vote



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

k-Nearest Neighbors

- Defer the decision to generalize beyond the training examples till a new query is encountered
  - Whenever we have a new point to classify, we find its K nearest neighbors from the training data.
  - The distance is calculated using some measure

k-Nearest Neighbors

- Given a query item:

  - Find k closest matches in a labeled dataset

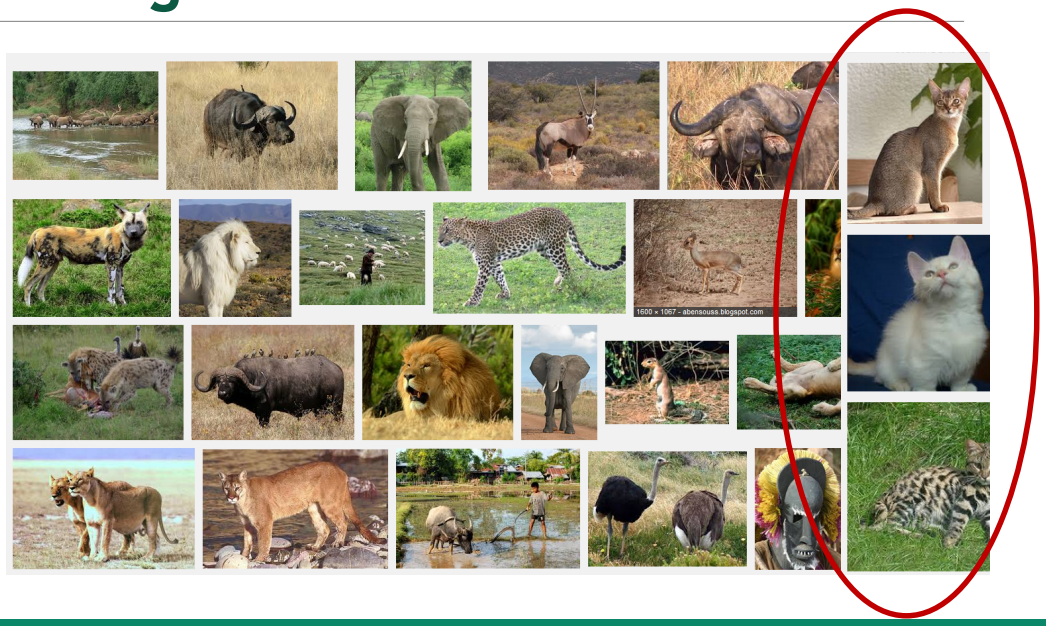
k-Nearest Neighbors

- Given a query item:

  - Return the most frequent label

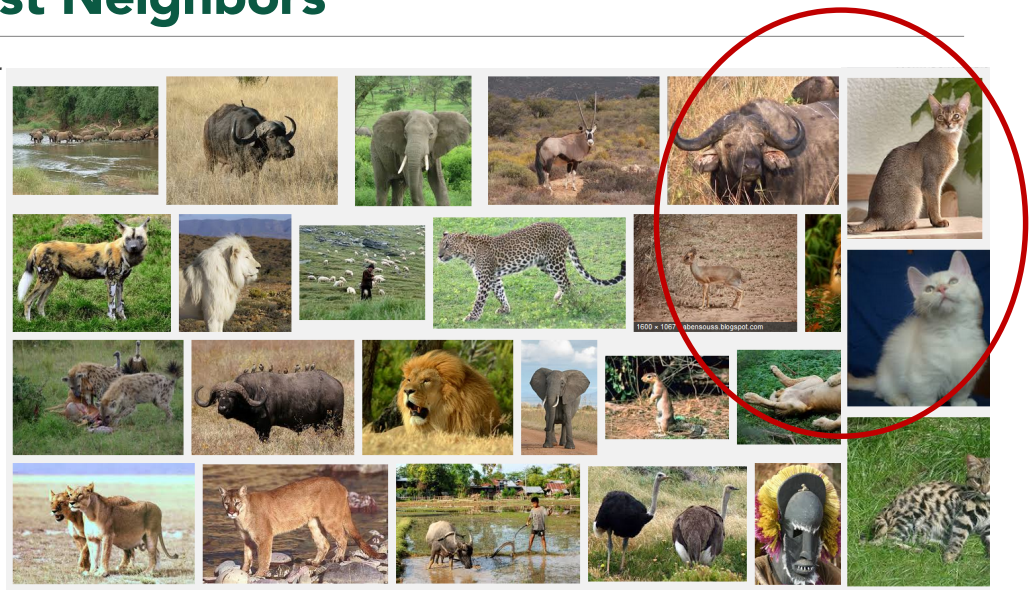
k-Nearest Neighbors

- k = 3 votes for "cat"



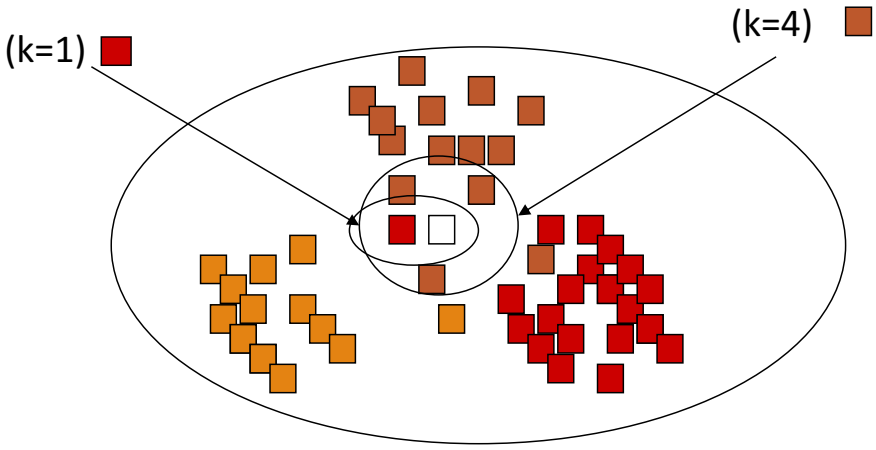
k-Nearest Neighbors

- 2 votes for cat,
- 1 each for Buffalo, Cat wins…
- Deer, Lion



K Nearest Neighbour (kNN) Classifier

- How many neighbors should we count ?

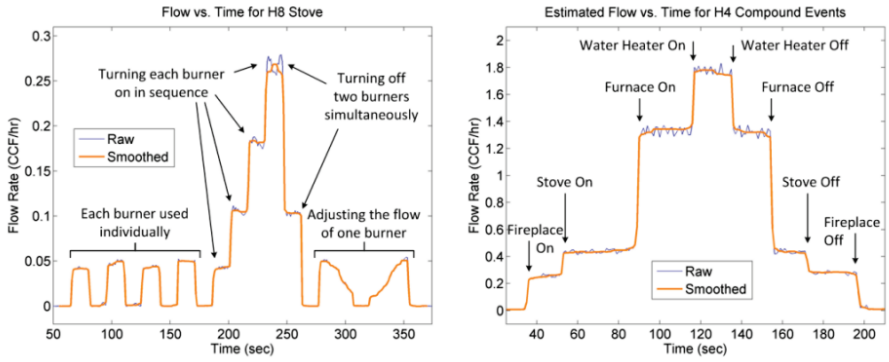


k-NN issues

- The Data is the model
  - No training needed.
  - Accuracy generally improves with more data.
  - Matching is simple and fast (and single pass).
  - Usually need data in memory, but can be run off disk.
- Minimal configuration:
  - Only parameter is k (number of neighbors)
  - Two other choices are important:
    - Weighting of neighbors (e.g. inverse distance)
    - Similarity metric

Application: Gas activity sensing

- Goal: Identify the appliance that is turned on/off
- Form a feature vector with (1) step size of flow change (2) time of change



k-NN Flavors

- Classification:
  - Model is y = f(X), y is from a discrete set (labels).
  - Given X, compute y = majority vote of the k nearest neighbors.
  - Can also use a weighted vote of the neighbors.
- Regression:
  - Model is y = f(X), y is a real value.
  - Given X, compute y = average value of the k nearest neighbors.
  - Can also use a weighted average of the neighbors.
- *Weight function is usually the inverse distance.*

K-NN Metrics or Measures

- Euclidean Distance: Simplest, fast to compute
  - $d(x, y) = \|x - y\|$
- Cosine Distance: Good for documents, images, etc.
  - $d(x, y) = 1 - \frac{x \cdot y}{\|x\|\|y\|}$
- Jaccard Distance: For set data:
  - $d(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$
- Hamming Distance: For string data:
  - $d(x, y) = \sum_{i=1}^{n} (x_i \neq y_i)$

K-NN metrics

- Manhattan Distance: Coordinate-wise distance
  - $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
- Edit Distance: for strings, especially genetic data.
- Mahalanobis Distance: Normalized by the sample covariance matrix – unaffected by coordinate transformations.

Choosing k

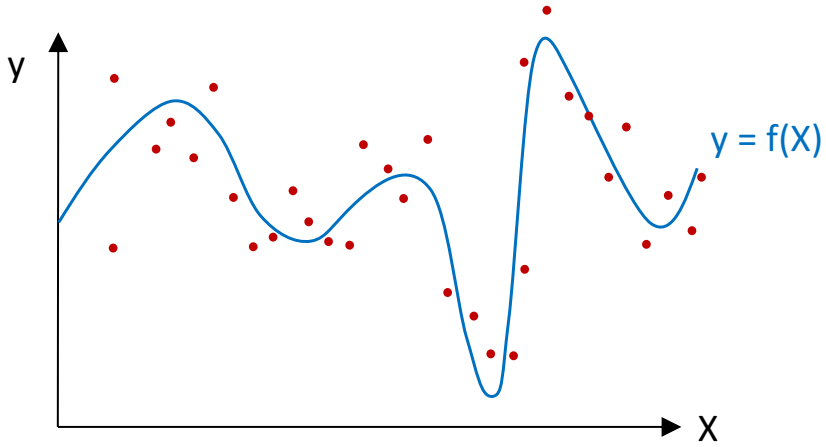
- Recall that prediction errors can be decomposed into two main categories:
  - Error due to *bias*
  - Error due to *variance*
- There is a tradeoff between a model's ability to minimize bias and variance
- There is a bias/variance tradeoff, depending on the value of K:
  - Small k → ?
  - Large k → ?
- Understanding these two types of error can help us diagnose model results and avoid the mistake of over- or under-fitting

Choosing k

- Error due to Bias
  - Difference between the expected (or average) prediction of our model and the correct value which we are trying to predict
- Error due to Variance
  - The variability of a model prediction for a given data point
- We have a bias/variance tradeoff
  - Small k → low bias, high variance
  - Large k → high bias, low variance

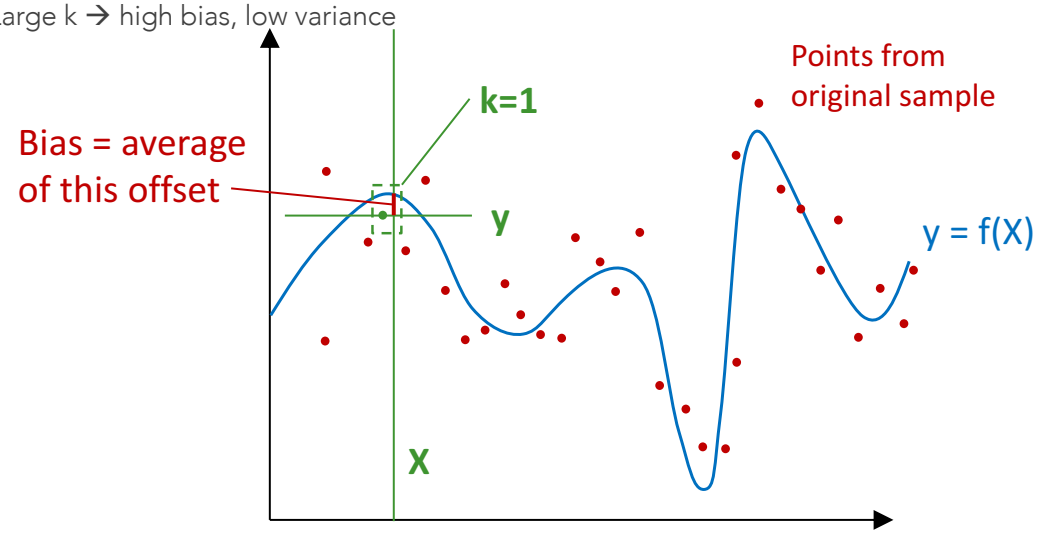
Choosing k

- Assume the real data follows the blue curve, with some mean-zero additive noise. Red points are a data sample.



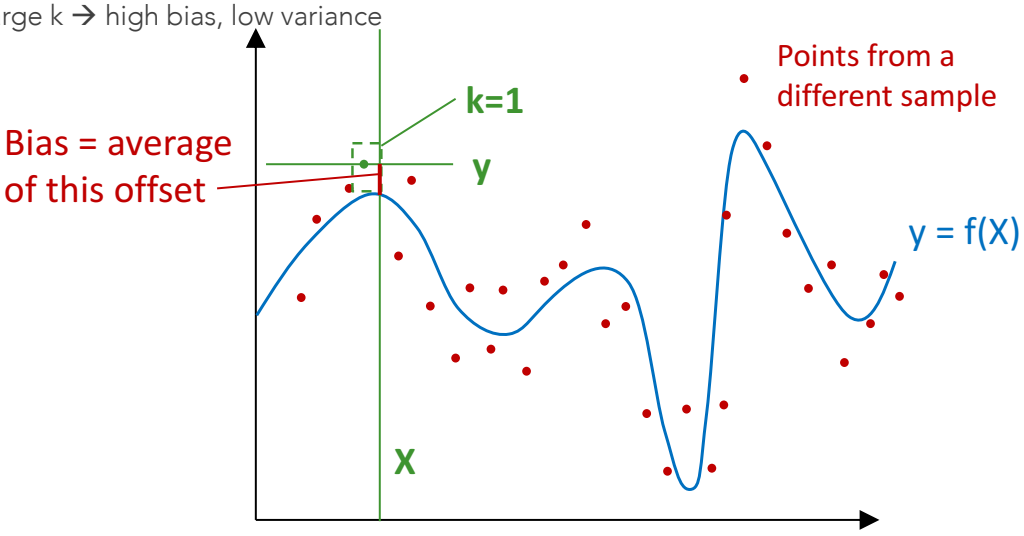
Choosing k

- Small k → low bias, high variance
- Large k → high bias, low variance



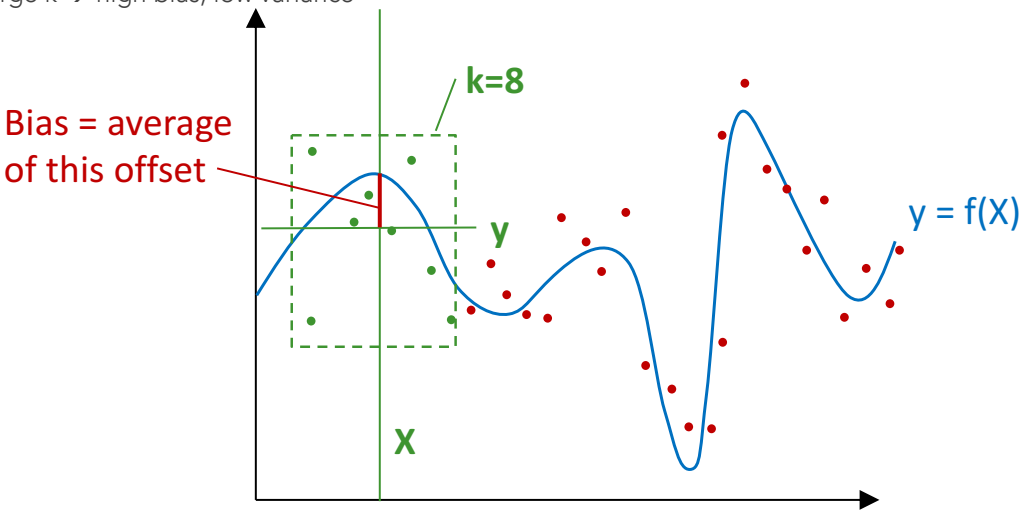
Choosing k

- Small k → low bias, high variance
- Large k → high bias, low variance



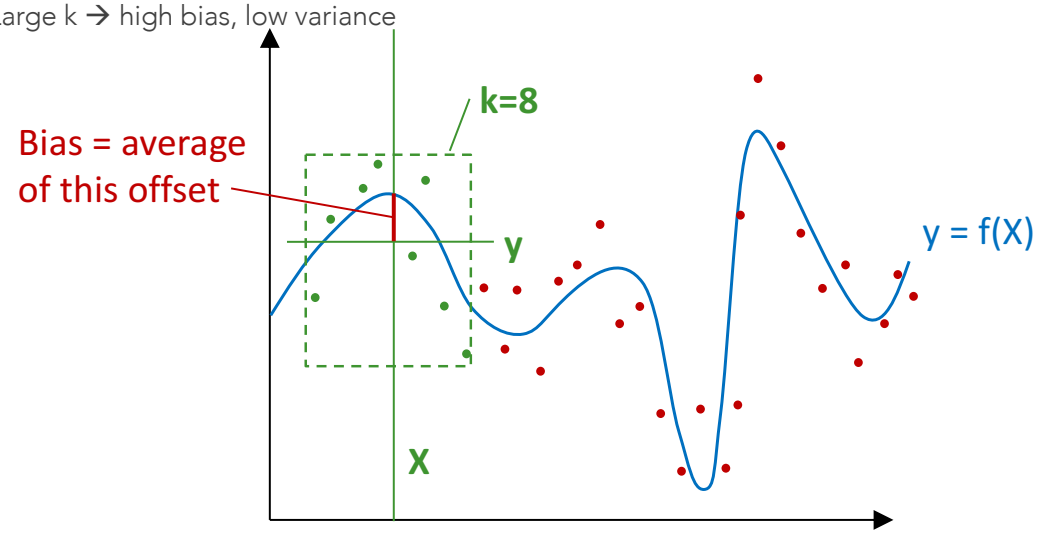
Choosing k

- Small k → low bias, high variance
- Large k → high bias, low variance



Choosing k

- Small k → low bias, high variance
- Large k → high bias, low variance



Choosing k in Practice

- Use cross-validation
  - Break data into train, validation and test subsets, e.g. 60-20-20% random split.
- Predict
  - For each point in the validation set, predict using the k-Nearest neighbors from the training set
  - Measure the error rate (classification) or squared error (regression).
- Tune
  - Try different values of k, and use the one that gives minimum error on the validation set.
- Evaluate
  - Test on the test set to measure performance

kNN and the Curse of Dimensionality

- The curse of dimensionality refers to phenomena that occur in high dimensions (100s to millions) that do not occur in low-dimensional (e.g. 3-dimensional) space.
- In particular data in high dimensions are much sparser (less dense) than data in low dimensions.
- For kNN, that means there are less points that are very close in feature space (very similar), to the point we want to predict

kNN and the Curse of Dimensionality

- Example: Consider a collection of uniformly random points in the unit cube. In one dimension, the average squared Euclidean distance between any two points is:
  - $\int_0^1 \int_0^1 (x-y)^2 dx\, dy = \frac{1}{6}$
- In N dimensions, we add up the squared differences for all N coordinates (because the coordinates are independent in a uniform random cube), giving:
  - $d^2 = \mathrm{E}[\|x-y\|^2] = \frac{N}{6}$
- So the Euclidean distance scales as $\sqrt{N}$

kNN and the Curse of Dimensionality

- From this perspective, its surprising that kNN works at all in high dimensions.
- Luckily real data are not like random points in a high-dimensional cube. Instead they live in dense clusters and near much lower-dimensional surfaces.
- Finally, points can be very "similar" even if their Euclidean distance is large. E.g. documents with the same few dominant words (with tf-idf weighing) are likely to be on the same topic.

When to Consider Nearest Neighbor ?

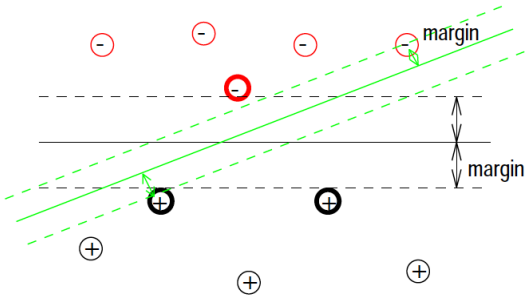
- Lots of training data
- Less than 20 attributes per example

Pros and Cons of K-NN

- Pros
  - Simple. No complex parameter tuning
  - No training is needed
  - Training is very fast
  - Learn complex target functions
  - Don't lose information
- Cons
  - Computationally expensive. Need to scan through all training samples
  - Slow at query time
  - Easily fooled by irrelevant attributes

Support Vector Machine

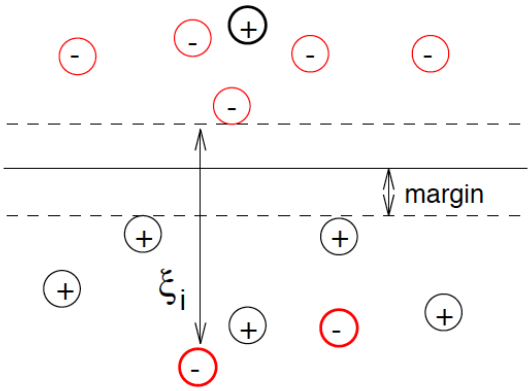
- Find a linear decision boundary that has the lowest generalization error
- Idea: the best one should have the largest margin



Note that this separating plane is only determined by the "support vectors"

Most of the time data are not linearly separable

- Include a penalty from mis-classification



Regularization parameter to weight training error

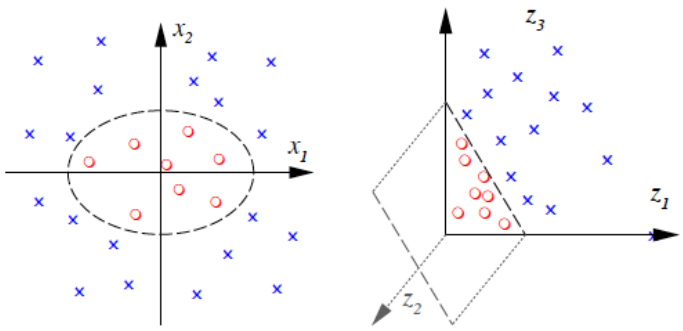
$$\min \; ||w||^2 + C\sum_{i=1}^{m} \xi_i$$

1/Margin

Classification error in training set

Non-linear SVM

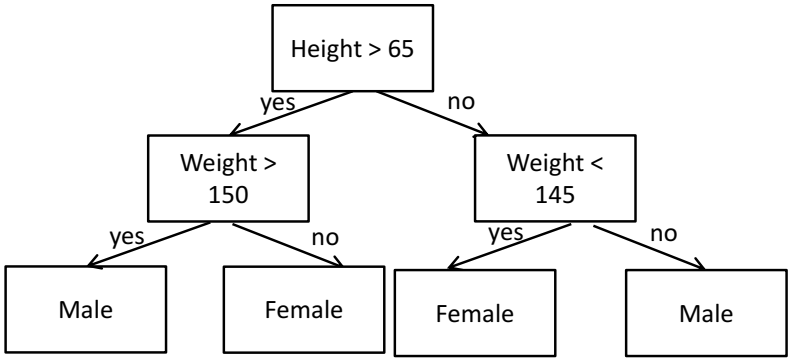
- Map data points to a high-dimensional space and do linear classification there
- Kernel trick: turns out solving SVM one only needs to known the feature space simil… …xplicitly do the mappin…

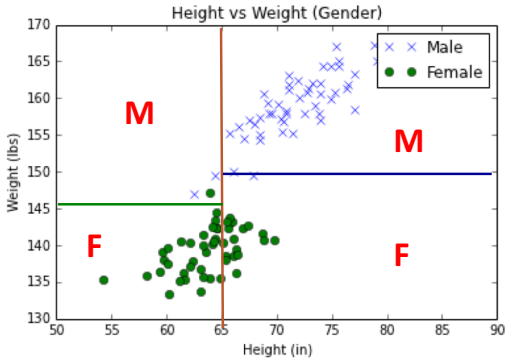


Decision Trees

- We have seen these before with Xbox Kinect and Motion Tracking
- Tree structure where nodes are "questions" and the leaves determine a classification or labeling

Decision Trees





Decision Trees

- Generally nonparametric, depending on learning algorithm, don't need to define depth or number of nodes
- Goal of learning algorithm is to find features that provide the most discrimination based on class
  - For example for determining gender, asking someone's height will provide pretty low error rate where something like eye color provides little information
- Algorithm Sketch:
  - Find feature that provides best separation accuracy
  - Create a node and separate data according to the feature
  - If zero error (all examples with height > 72 are male, make a leaf node)
  - Else: recurse for each subset generated.
- Common Algorithm is ID3, a precursor to the C4.5 algorithm

Decision Trees

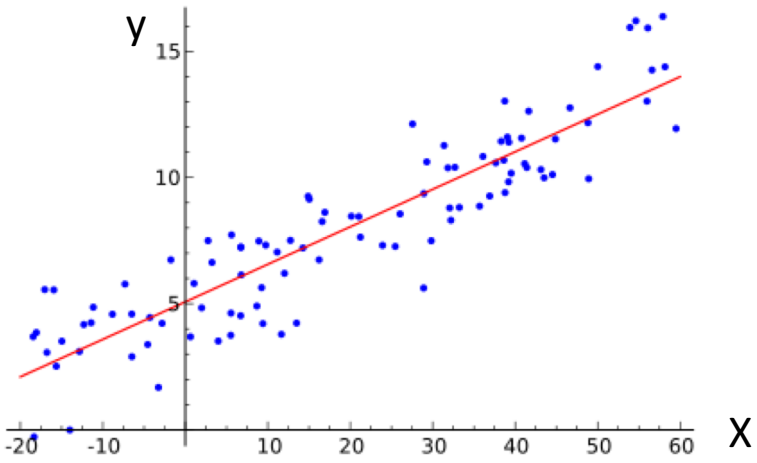
- Advantages:
  - Model is very easy to interrogate, you can look at each node and easily understand its purpose (i.e. the feature and the threshold or value)
  - Can handle multi-domain data (real numbers, categorical data, all together)
  - Cost of classification is O(log n) where n in the tree depth
- Disadvantages:
  - Optimization is most likely full of local optima and is an NP-complete problem (i.e. exact computation is too expensive)
  - Highly sensitive to the distribution of the training data (often over-fits!)

Logistic Regression

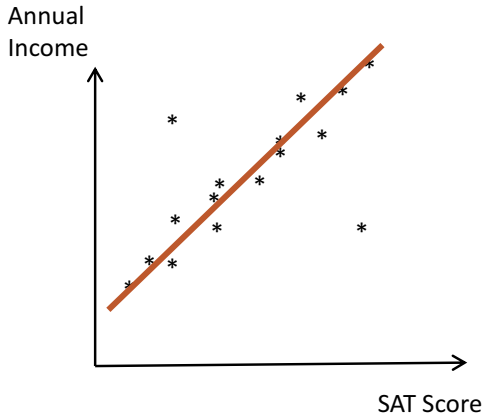
- Logistic regression is probably the most widely used general-purpose classifier.
- Its very scalable and can be very fast to train. It's used for
  - Spam filtering
  - News message classification
  - Web site classification
  - Product classification
  - Most classification problems with large, sparse feature sets.
- The only caveat is that it can overfit on very sparse data, so its often used with *regularization*

Linear Regression

- We want to find the best line (linear function y=f(X)) to explain the data.



Linear Regression



Predict Alice's annual income given her SAT score

Model: straight line

$y = c_1 x + c_2$

Parameters

Least squares solution!

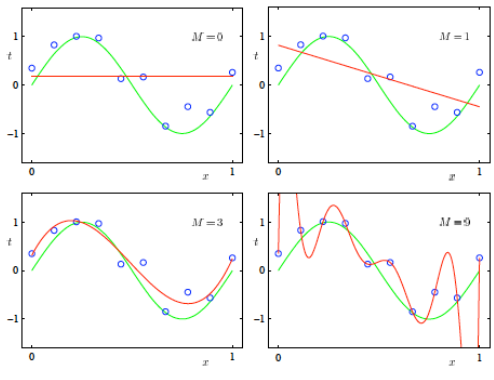
High-Order Polynomial Models

- If data has more complicated structure, we may use high-order polynomial models

Polynomial curve fitting

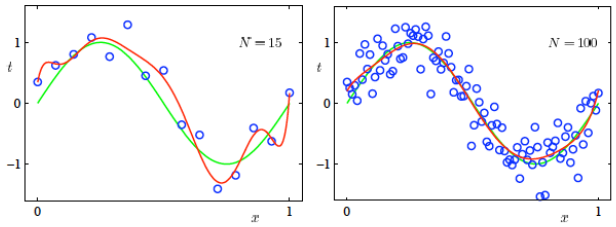
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} [y(x_n, \mathbf{w}) - t_n]^2$$



More Data are Needed to Learn Correct Model

More data are needed to learn correct model



This is *overfitting*.

example from Bishop (2006), *Pattern Recognition and Machine Learning*

You need a lot more data to learn a degree-100 polynomial

Logistic Regression

- Logistic regression is designed as a binary classifier (output say {0,1}) but actually outputs the probability that the input instance is in the "1" class.

- A logistic classifier has the form:

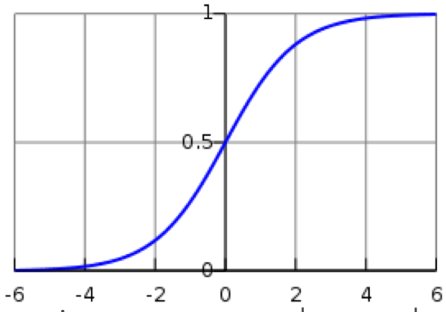
$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

where $X = (X_1, \ldots, X_n)$ is a vector of features.

Logistic Regression

- Logistic regression maps the "regression" value $-X\beta$ in (-∞,∞) to the range [0,1] using a "logistic" function:

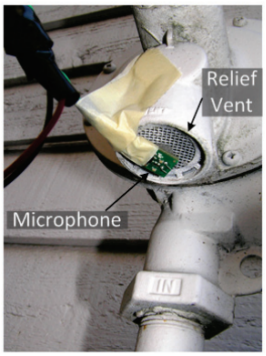
$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$



- i.e. the logistic function maps any value on the real line to a probability in the range [0,1]

Application: Gas activity sensing

- Predict the amount of gas flow using microphone

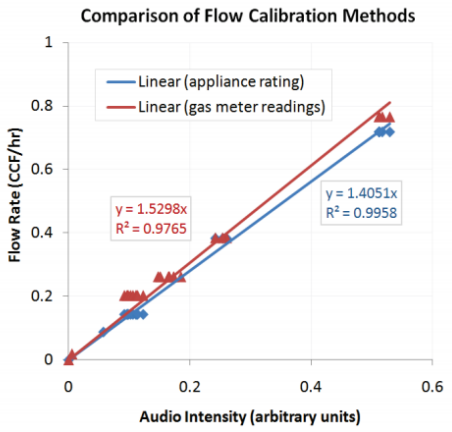


Exploit the "hissing sound" when gas pass through the gas regulator

[Cohn et al, Pervasive 2010]

Audio intensity is linear to gas flow

- A simple linear regression does the job



Logistic Regression: Summary

- What is in the model?
  - Linear separating plane
  - Logistic function for probability
- Summary
  - Training is fast
  - Very simple model (less overfitting)
  - Prediction is also fast
  - The output of prediction has simple explanation (probability of belonging to a class)

Recap: Comparison of Classifiers

	Decision Tree	Logistic Regression	SVM
Model	Tree (horizontal/vertical separating plane)	Linear separating plane	Can also use non-linear separating plane
Simplicity	Very simple	Simple	Complicated
Interpretation of output	Clear, follow the questions asked	Probability	Unclear
Chances of over-fit	High	Less worry	Should be careful
Development time	Just plug data in	Just plug data in	Tuning tuning tuning
Off-the-shelf accuracy	Okay	Okay	Probably the highest
Use case	When features are clear	Often combined with neural networks	Often suggest RBF as a starting point if you have zero knowledge on data

Random Forests

- Grow K trees on datasets sampled from the original dataset with replacement (bootstrap samples), p = number of features.
- Draw K bootstrap samples of size N
- Grow each Decision Tree, by selecting a random set of m out of p features at each node, and choosing the best feature to split on.
- Aggregate the predictions of the trees (most popular vote) to produce the final class.
- Typically m might be e.g. sqrt(p) but can be smaller.

Random Forests

- Principles: we want to take a vote between different learners so we don't want the models to be too similar. These two criteria ensure diversity in the individual trees:
- Draw K bootstrap samples of size N:
  - Each tree is trained on different data.
- Grow a Decision Tree, by selecting a random set of m out of p features at each node, and choosing the best feature to split on.
  - Corresponding nodes in different trees (usually) cant use the same feature to split.

Random Forests

- Very popular in practice, probably the most popular classifier for dense data (<= a few thousand features)
- Easy to implement (train a lot of trees). Good match for MapReduce.
- Parallelizes easily (but not necessarily efficiently).
- Not quite state-of-the-art accuracy – boosted trees generally do better – or DNNs.
- Needs many passes over the data – at least the max depth of the trees. (<< boosted trees though)
- Easy to overfit – hard to balance accuracy/fit tradeoff.

Clustering – Why?

Clustering has one or more goals:

- Segment a large set of cases into small subsets that can be treated similarly - segmentation
- Generate a more compact description of a dataset - compression
- Model an underlying process that generates the data as a mixture of different, localized processes – representation

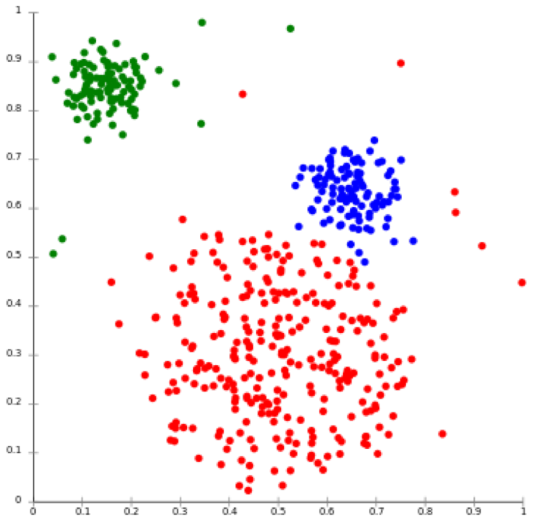
Clustering – Why?

Examples:

- Segment: image segmentation
- Compression: Cluster-based kNN, e.g. handwritten digit recognition.
- Underlying process: Accents of people at some place – because place of origin strongly influences the accent you have.

Stereotypical Clustering

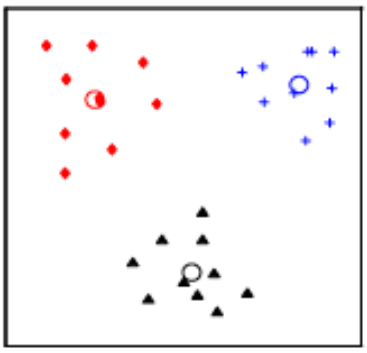
- Note: Points are samples plotted in feature space, e.g. 10,000-dimensional space for 100x100 images.



K-means

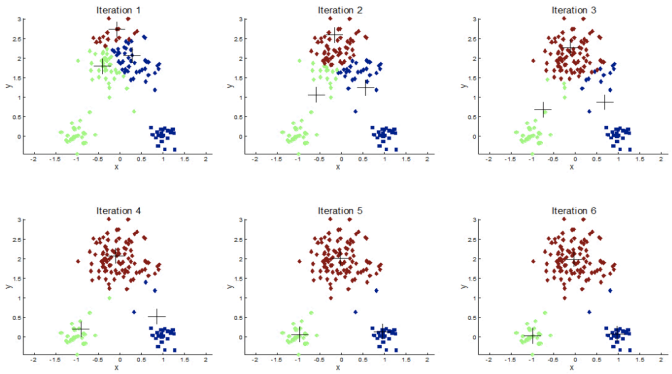
- Very popular algorithms for clustering
- Minimize the sum of distances to cluster centroids

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2.$$



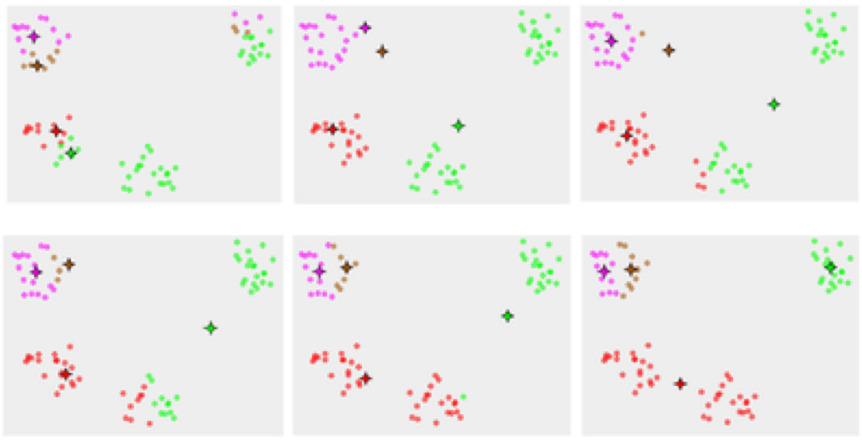
K-means: Algorithm

1. Compute the centroids of each class
2. Update each example to the closest cluster centroid.

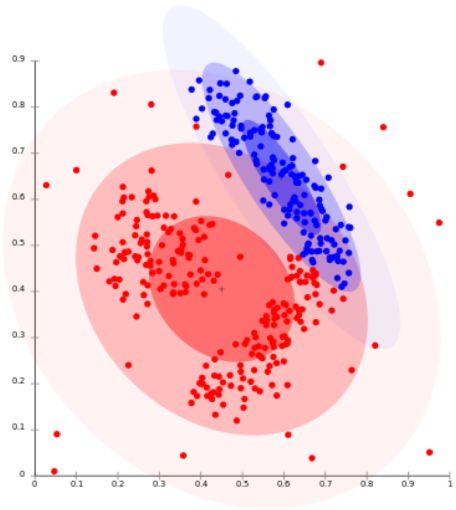


K-means May Be Stuck at Local Optimal

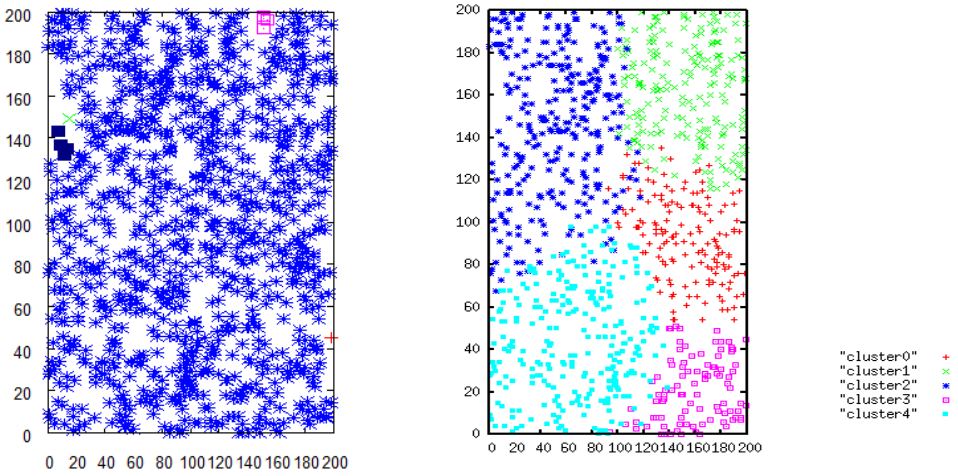
- Centroid initialization is important



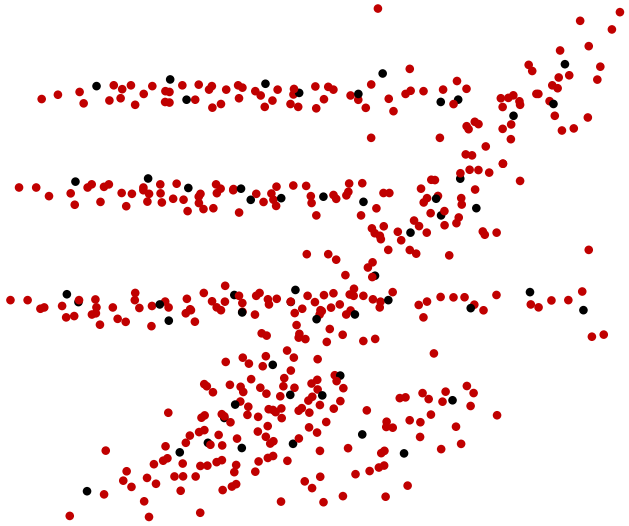
Model-based Clustering



Clustering for Segmentation



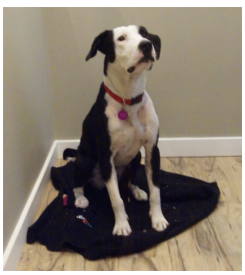
Condensation/Compression



"Cluster Bias"

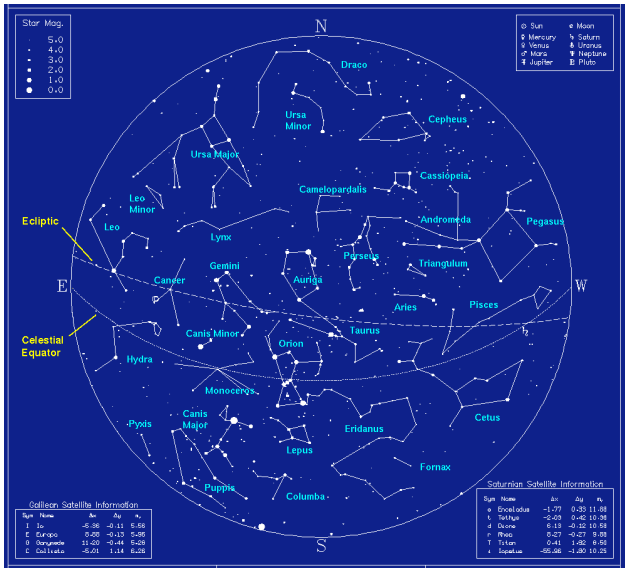
- Human beings conceptualize the world through categories represented as examplars (Rosch 73, Estes 94).





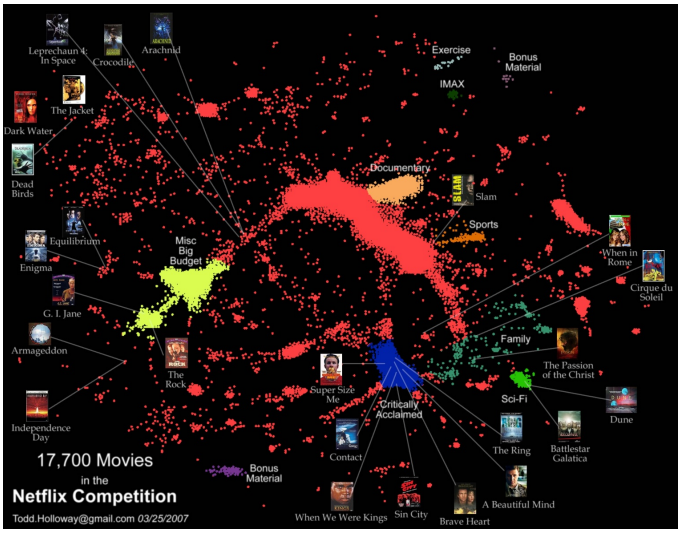
- We tend to see cluster structure whether it is there or not.
- Works well for dogs, but…

Cluster Bias



Netflix

- More of a continuum than discrete clusters
- Factor models, kNN do much better than discrete cluster models.



"Cluster Bias"

- Upshot
  - Clustering is used more than it should be, because people assume an underlying domain has discrete classes in it.
  - This is especially true for characteristics of people, e.g. Myers-Briggs personality types like "ENTP".
  - In reality the underlying data is usually continuous.
  - Just as with Netflix, continuous models (dimension reduction, kNN) tend to do better.

Terminology

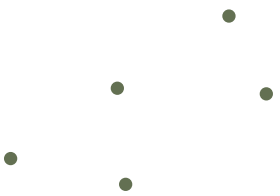
- Hierarchical clustering: clusters form a hierarchy. Can be computed bottom-up or top-down.
- Flat clustering: no inter-cluster structure.
- Hard clustering: items assigned to a unique cluster.
- Soft clustering: cluster membership is a real-valued function, distributed across several clusters.

K-means clustering

- The standard k-means algorithm is based on Euclidean distance.
- The cluster quality measure is an intra-cluster measure only, equivalent to the sum of item-to-centroid kernels.
- A simple greedy algorithm locally optimizes this measure (usually called Lloyd's algorithm):
  - Find the closest cluster center for each item, and assign it to that cluster.
  - Recompute the cluster centroid as the mean of items, for the newly-assigned items in the cluster.

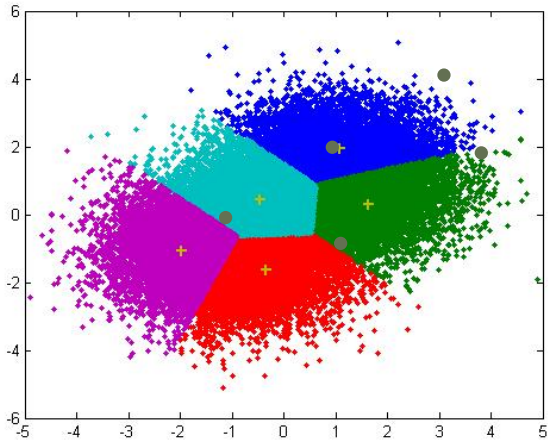
K-means clustering

- Cluster centers – can pick by sampling the input data.



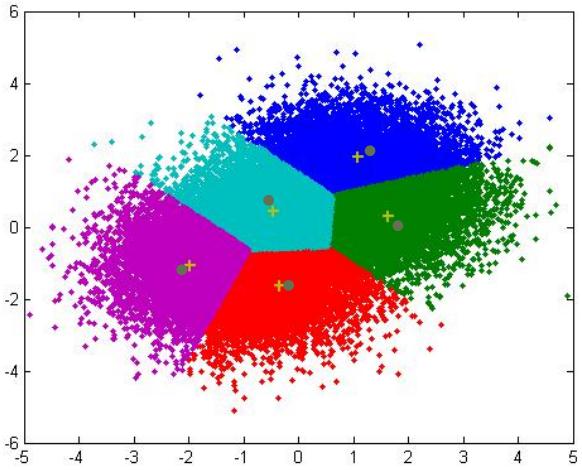
K-means clustering

- Assign points to closest center



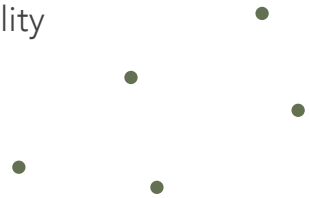
K-means clustering

- Recompute centers (old = cross, new = dot)



K-means clustering

- Iterate
  - For fixed number of iterations
  - Until no change in assignments
  - Until small change in quality



K-means properties

- It's a greedy algorithm with random setup – solution isn't optimal and varies significantly with different initial points.
- Very simple convergence proofs.
- Performance is O(nk) per iteration, not bad and can be heuristically improved.
  n = total features in the dataset, k = number clusters
- Many generalizations, e.g.
  - Fixed-size clusters
  - Simple generalization to m-best soft clustering
- As a "local" clustering method, it works well for data condensation/compression.

Choosing clustering dimension

- AIC or Akaike Information Criterion:

$$\textbf{AIC:} \qquad K = \arg\min_{K}[-2L(K) + 2q(K)]$$

- K=dimension, L(K) is the likelihood (could be RSS) and q(K) is a measure of model complexity (cluster description complexity).
- AIC favors more compact (fewer clusters) clusterings.
- For sparse data, AIC will incorporate the number of non-zeros in the cluster spec. Lower is better.

K-means

- Advantages
  - Simple, easy to implement
  - Very parallelizable
  - Easy to extend to use other distance metrics
- Disadvantages
  - Sensitive to initial approximation
  - Easily stuck at local optimal especially when dimensionality is high
  - May be difficult to determine the number of clusters

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

5-minute break

Unsupervised Learning Can Help Supervise Learning

- Obtaining labeled examples is labor intensive
- Can we teach the machines to learn some structures from unlabeled samples, and then exploit the structures for classification?
- This is semi-supervised learning

Outline for this Evening

- Three Basic Algorithms
  - kNN
  - Linear Regression
  - K-Means
- Training Issues
  - Measuring model quality
  - Over-fitting
  - Cross-validation

Model Quality

- Almost every model optimizes some quality criterion:
  - For linear regression it was the Residual Sum-of-Squares
  - For k-Means it is the "Inertia" – the mean squared distance from each sample to its cluster center.
  - …
- The quality criterion is chosen often because of its good properties:
  - Convexity: so that there is a unique, best solution
  - Closed form for the optimum (linear regression) or at least for the gradient (for SGD).
  - An algorithm that provably converges.

Model Quality

- There are typically other criteria used to measure the quality of models. e.g. for clustering models
  - Silhouette score
  - Inter-cluster similarity (e.g. mutual information)
  - Intra-cluster entropy
- For regression models
  - Stability of the model (sensitivity to small changes)
  - Compactness (sparseness or many zero coefficients)

Evaluating Clusterings: Silhouette

- The silhouette score is

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

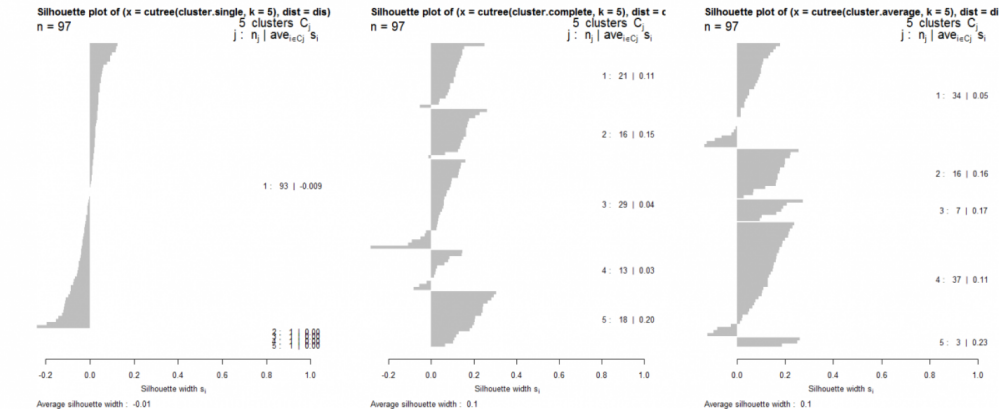
  - where a(i) is the mean distance from sample i to its own cluster,
  - b(i) the mean distance from i to the second-closest cluster.
  - -1 ≤ S(i) ≤1
- Provides a succinct graphical representation of how well each object lies within its cluster
- Perhaps surprisingly, silhouette scores can be, and often are, negative

Evaluating Clusterings: Silhouette

- silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters
- A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters
- Negative values indicate that those samples might have been assigned to the wrong cluster.

Evaluating Clusterings: Silhouette

- Silhouette plot: horizontal bars with cluster score.
- Sort (vertically) first by cluster, then by score.



Regularization with Secondary Criteria

- While secondary criteria can be measured after the model is built, its too late then to affect the model.
- Using secondary criteria during the optimization process is called "regularization".
- Examples:
  - L1 (LASSO) regularization adds a term to the measure being optimized which is the sum of absolute value of model coefficients.
  - L2 (Ridge) regularization adds a term to the measure being optimized which is the sum of squares of model coefficients.

Regularization with Secondary Criteria

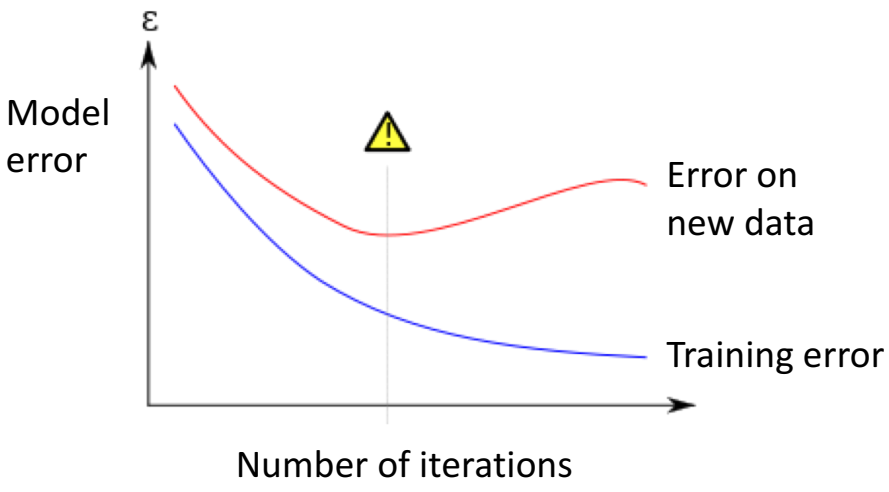
- L1 regularization in particular is very widely used. It has the following impacts:
  - Yields a convex optimization problem in many cases, so there is a unique solution.
  - The solution is usually stable to small input changes.
  - The solution is quite sparse (many zero coefficients) and requires less disk and memory to run.
  - L1 regularization on factorization models tends to decrease the correlation between model factors.

Over-fitting

- Your model should ideally fit an infinite sample of the type of data you're interested in.
- In reality, you only have a finite set to train on. A good model for this subset is a good model for the infinite set, up to a point.
- Beyond that point, the model quality (measured on new data) starts to decrease.
- Beyond that point, the model is **over-fitting** the data.

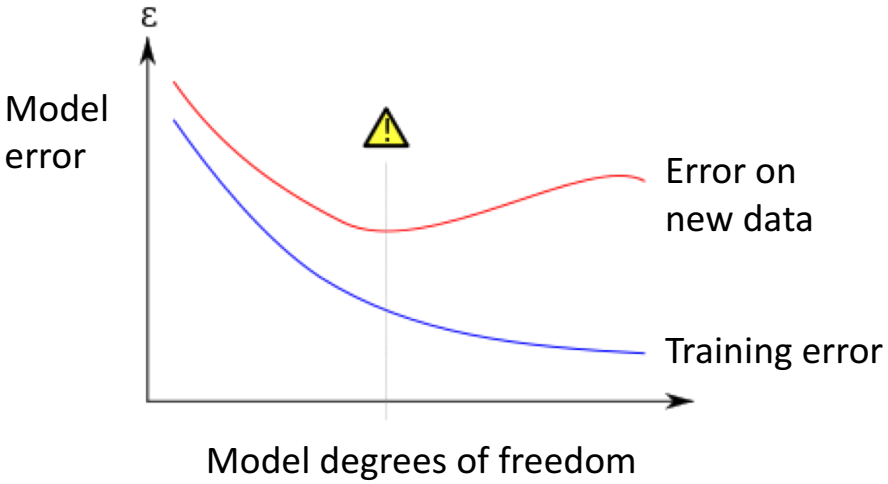
Over-fitting

- Over-fitting during training



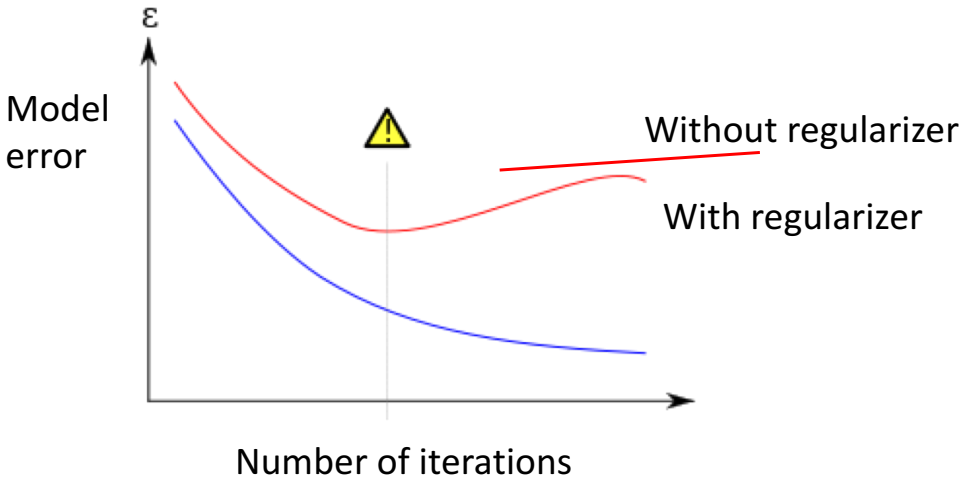
Over-fitting

- Another kind of over-fitting



Regularization and Over-fitting

- Adding a regularizer:



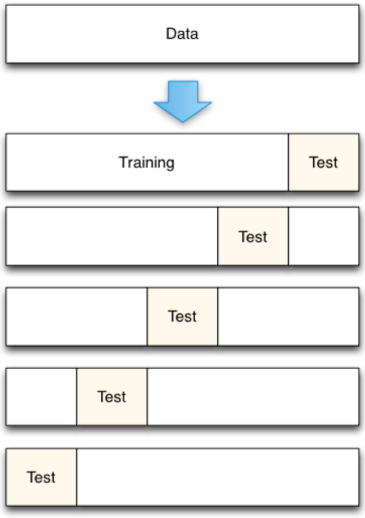
Cross-Validation

- Cross-validation involves partitioning your data into distinct training and test subsets.
- The test set should never be used to train the model.
- The test set is then used to evaluate the model after training.

K-fold Cross-Validation

- To get more accurate estimates of performance you can do this k times.
- Break the data into k equal-sized subsets A_i
- For each i in 1,…,k do:
  - Train a model on all the other folds A_1,…, A_{i-1}, A_{i+1},…, A_k
  - Test the model on A_i
- Compute the average performance of the k runs

5-fold Cross-Validation



Summary

- Three Basic Algorithms
  - kNN
  - Linear Regression
  - K-Means
- Training Issues
  - Measuring model quality
  - Over-fitting
  - Cross-validation